WILLKIE FARR & GALLAGHER LLP

787 Seventh Avenue
New York, NY 10019-6099
(212) 728-8757

FAX TRANSMISSION

Date: September 15, 2005 Time: | Total number of pages (including this page): 23

Please include Client/Matter No. below

FROM: David E. Boundy

Room No.: 4578
Phone No.: (212) 728-8757
Direct FAX: (212) 728-9757

TO: Art Unit 2183, SPRE Shop	City:	Alexandria	State:	Virginia
U.S. Patent and Trademark Office	Fax No.:	571 273 8300	Telephone No.:	571-272-2100
	City:	Alexandria	State:	Virginia

CONCERNING APPLICATION:

Applicant(s):	John S. Yates, Jr., et al.
Serial No.:	09/626,325
Filed:	July 26, 2000
Title:	OPERATING SYSTEM FOR COMPUTER WITH TWO ARCHITECTURES
Art Unit:	2183
Examiner:	R. Ellis

AFTER FINAL – EXPEDITED PROCEDURE

I hereby certify that the attached

- This FAX cover sheet
- Petition for Review by Technology Center SPRE (without exhibits)

along with any paper(s) referred to as being attached or enclosed) are being transmitted by facsimile on September 15, 2005 to Art Unit 2183, SPRE Shop, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dated: September 15, 2005

By: DWE Boundy
David E. Boundy
Registration No. 36,461

Attention Recipient:
If Any Problems: Call (212) 728-8911
Receiving Fax Number: (212) 728-8111

05 SEP 15 PM 6:04

Internal Use Only:
Client No.: 114596-28-000053BS Matter No.: Attorney No.: 12256
Please check here if you want faxed document returned to you *instead* of sent to Records Department.

PATENT

ATTORNEY DOCKET NO. 114596-28-000053BS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial No.:	09/626,325	Confirmation No.:	7939
Applicant:	John S. Yates, Jr., et al.		
Title:	OPERATING SYSTEM FOR COMPUTER WITH TWO ARCHITECTURES		
Filed:	July 26, 2000		
Art Unit:	2183		
Examiner:	R. Ellis		
Atty. Docket:	114596-28-000053BS		
Customer No.	38492		

AFTER FINAL – EXPEDITED PROCEDURE

PETITION FOR REVIEW BY TECHNOLOGY CENTER SPRE

Art Unit 2183, SPRE Shop
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Petitioner requests review by the Director's representative in the Technology Center of the following two issues that arise out of the Office Action of July, 19, 2005:

1. Did the examiner err in objecting to the "Amendment Accompanying RCE" of April 27, 2005 as "new matter?"

Yes. An amendment need not have literal support in the specification, as the examiner requires; "inherent" support is entirely sufficient. Here, the amendment to the specification merely states the "ordinary meaning" in the art of a term, and the ordinary theory of operation, and thus is not new matter.

I certify that this correspondence, along with any documents referred to therein, is being transmitted by facsimile, without exhibits, on September 15, 2005 to Art Unit 2183 at FAX no. 571 273 8300, and deposited with the United States Postal Service, with exhibits, on September 15, 2005 as First Class Mail in an envelope with sufficient postage addressed to Art Unit 2183, SPRE Shop, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

2. May this first Office Action after an RCE be made final?

No. Examination is incomplete – that which is incomplete cannot be final. An Information Disclosure Statement that was timely filed and cited in the Request for Continued Examination has not been considered. The examiner states that he disregards the effect of the amendment because of the "new matter" issue; the MPEP instructs otherwise. Until the application has been timely and completely examined in the manner required by the Director, final rejection is premature.

This Petition is timely presented within two months of the examiner's action of July 19, 2005.

I. The Amendment of April 27, 2005 Adds No "New Matter"

The Office Action of July 19, 2005 "objects" to an amendment to the specification proposed in the "Amendment Accompanying RCE" of April 27, 2005. There is no "new matter" or § 112 ¶ 1 rejection of any claim. Thus, MPEP § 608.04(c) provides that the "objection" is reviewable by petition.

A. The Text of the Amendment

The amendment to which the examiner "objects" is the underlined portion of the following. As will be shown below, the underlined language falls into three categories recognized not to be "new matter:" (a) legal principles of interpretation that are inherent in every patent application, and (b) the ordinary and customary meanings of several terms, as those terms are understood and actually used by those of skill in the relevant art, inherent in the use of the terms themselves, and (c) the ordinary theory of operation of "processes" and "thread," which is also implicit in the terms themselves:

> Referring to **Figs. 3a** and **3l** and to Table 1, X86 threads (*e.g.*, **302, 304**) managed by X86 operating system **306**, carry the normal X86 context, including the X86 registers, as represented in the low-order halves of r32-r55, the EFLAGS bits that affect execution of X86 instructions, the current segment registers, etc. (The terms "process" and "thread" are used herein in their ordinary and customary, though formal, senses, as actually used in the programming language systems, operating systems, and processor architecture arts. Generally, a "process" is a unit of processor scheduling and protection, each with an associated data structure (or set of data structures) that, in most implementations, holds machine register values and other context associated with the process. The process data structures, and thus the processes of a computer, are usually under the management of an operating system, usually the operating system's scheduler. Generally, a "thread" is a flow of control within a process. Each thread has an associated data structure (or set of data structures)

> that, in most implementations, hold machine register values (usually different than the registers associated with a process) and other context associated with the thread. The thread data structures, and thus the threads of a process, are usually managed either by an operating system or other run time system, to permit the thread to be scheduled independently of and concurrently with other threads of the same process.) In addition, if an X86 thread **302, 304** calls native Tapestry libraries **308,** X86 thread **302, 304** may embody a good deal of extended context, the portion of the Tapestry processor context beyond the content of the X86 architecture. A thread's extended context may include the various Tapestry processor registers, general registers r1-r31 and r56-r63, and the high-order halves of r32-r55 (see Table 1), the current value of ISA bit **194** (and in the embodiment of section IV, *infra*, the current value of XP / calling convention bit **196** and semantic context field **206**).

This amendment was made to resolve a dispute relating to the term "thread." For nearly two years, Applicant has encouraged the examiner to interpret the term "thread" in its ordinary and customary, though formal, sense, as actually used in the arts relevant to this application: programming language systems, operating systems, and processor architecture. As noted below, the precise definition of "thread" varies somewhat vendor-to-vendor, but is consistent in one respect: every relevant dictionary and technical reference that states a formal definition of the term "thread" emphasizes that "threads" provide one or more concurrent or independent flows of control.[1]

In contrast, the examiner relies on an extrapolation of his own devising, a definition that is not found in any dictionary or any other source, and for which he has been unable to provide a single example of actual use in the art. The underlying sources from which he extrapolates are obsolete, and provide only irrelevant general definitions, rather than the definition that is specific to the relevant arts. The examiner defines "thread" as any "process that is part of a larger process or program," including anything that "handles an interrupt" (Office Action of 10/27/2004, page 2, last 3 lines), even if the handler is structured specifically to prevent concurrent execution.

---

[1] As will be shown below, the art allows for single-thread processes, in the simplest case. "Concurrent" in the relevant arts "[pertains] to the occurrence of two or more activities within the same interval of time, achieved either by interleaving the activities or by simultaneous execution. *Synonym*: parallel. *Contrast*: simultaneous." Authoritative Dictionary of IEEE Standards Terms, 7th ed. (2000).

B. Every Definition of "Thread" In the Relevant Art Implies "Concurrency"

1. Vendor-Neutral Definitions of "Thread" and "Process" Uniformly Note that Threads Allow "Concurrency"

Various technical dictionaries give the following definitions of the terms "process" and "thread" in the specific context of the arts relevant to this application. Note that every single definition in the specific arts of this application is consistent in stating that "concurrency" (or "parallelism" or "independent scheduling") is the basic "reason for being" of threads.

The Authoritative Dictionary of IEEE Standards Terms, 7th ed. (2000) (Exhibit A) collects formal definitions from various formal standards issued by various official and non-governmental organization standards bodies. The IEEE Dictionary gives the following relevant definitions (underline added):

> **thread (4)** A single flow of control within a process. Each thread has its own thread ID, scheduling priority and policy, *errno* value, thread-specific key/value bindings, and the required system resources to support a flow of control. ...[2]
>
> **thread of control** A sequence of instructions executed by a conceptual sequential subprogram, independent of any programming language. More than one thread of control may execute concurrently, interleaved on a single processor, or on separate processors. The conceptual threads of control in an Ada application are Ada tasks. They may, but need not, correspond to the POSIX threads defined in POSIX.1.[3]

In definition "thread (4)," the idea of "concurrency" is inherent in the terms "scheduling priority and policy:" scheduling which of several threads is to run at any given time only makes sense if they can execute concurrently.

The examiner himself (Office Action of 10/27/2004, page 2) cited the following definition as his first choice definition, in the Office Action of 10/27/04. The examiner found this definition on the Yahoo! web site, and Yahoo attributes the definition to The American

---

[2] Definition (4) originated with the International Organization for Standardization and International Electrotechnical Commission, in their ISO/IEC Standard No. 9945-1-1996, Portable Operating System Interface (POSIX), the formal definition for UNIX operating systems.

The IEEE Dictionary gives another definition, "thread (3) A single sequential flow of control within a process." However, definition (3) is stated only in standards that relate to conformance testing of software developed to implement definition (4). Thus, concurrency is inherently part of definition (3) as well.

[3] The POSIX definition is definition (4).

Heritage Dictionary of the English Language: Fourth Edition (2000) (Exhibit B) (underline added):

> **thread**, n. *Computer Science* a. A portion of a program that can run independently of and concurrently with other portions of the program.

The "Wikipedia" web site is generally regarded as an accurate resource. Wikipedia defines "thread" as follows http://en.wikipedia.org/wiki/Thread_%28computer_science%29 (Exhibit C) (bold in original, underline added)[4]

> Many programming languages, operating systems, and other software development environments support what are called **"threads"** of execution. Threads are similar to processes, in that both represent a single sequence of instructions executed in parallel with other sequences, either by time slicing or multiprocessing. ...
>
> An advantage of a multi-threaded program is that it can operate faster on computer systems that have multiple CPUs, or across a cluster of machines. This is because the threads of the program naturally lend themselves for truly concurrent execution....
>
> Threads allow a program to do multiple things concurrently. ...
>
> **Implementations**
>
> There are many different and incompatible implementations of threading. These can either be kernel-level or user-level implementations. ...

Within months of the filing date of this application, the specialized definition of the term "thread" had become integrated into **freshman-level** computer science courses. For example, slides for a Fall 2000 freshman-level course (http://www.cse.ucsd.edu/classes/fa00/cse120/lectures/5-threads.pdf, Exhibit D) at the University of California, San Diego explains threads as follows (underline added):

- A thread is bound to a single process
  - Processes, however, can have multiple threads
- Threads become the unit of scheduling
  - Processes are now the containers in which threads execute
  - Processes become static, threads are the dynamic entities

[4] A printed copy of this definition from http://www.wordiq.com/definition/Thread_%28computer_science%29 was included with Applicant's paper of July 26, 2004.

Lecture notes from January 2000, http://www.andrew.cmu.edu/course/15-412/ln/412springlecture4.html (Exhibit E), describe the relationship of processes and threads that was taught at Carnegie-Mellon University. "Context switch" is another term for scheduling, for sharing execution among several threads or processes that are all ready to execute concurrently (italic and bold in original, underline added):

> **Threads, a.k.a Light-Weight Processes (LWP)**
>
> Now suppose that we want multiple process-like things that are separately schedulable -- but share memory. ...
>
> Now consider multiple "processes" sharing the same memory, but otherwise maintaining different state (registers, &c). We call these processes within processes *threads*. We call them this, because they are separate *threads of control* within the same process....
>
> - Fast context switch: ...
>
> ...
>
> The PCB now maintains the state of each thread and allows each thread to be scheduled independently. ...

2. Vendor-Specific Definitions of "Thread" Uniformly State a Requirement for "Concurrency"

By 1999, many different operating systems implemented some form of "threads." Definitions drawn from various manufacturers' documentation vary somewhat in detail, but are consistent in their emphasis on "concurrency," "parallelism," or "independent scheduling."

At its Microsoft Developer's Network (Microsoft's technical library directed to those who must have precise and accurate information, as opposed to "home users," the audience for the Microsoft Computer Dictionary), Microsoft describes "threads" as follows http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/about_processes_and_threads.asp, multiple_threads.asp and creating_threads.asp (Exhibit F) (italic and bold in original, underline added):

> **About Processes and Threads**
>
> A *thread* is the entity within a process that can be scheduled for execution. ... Microsoft® Windows® supports *preemptive multitasking*, which creates the effect of simultaneous execution of multiple threads from multiple processes. On a multiprocessor computer, the system can simultaneously execute as many threads as there are processors on the computer.

Creating Threads

The CreateThread function creates a new thread for a process. ... A process can have multiple threads simultaneously executing the same function.

Hewlett-Packard describes its implementation of threads in http://docs.hp.com/en/B2355-90695/pthread.3T.html (Exhibit G):

THREAD OVERVIEW

A *thread* is an independent flow of control within a process, composed of a context (which includes a register set and a program counter) and a sequence of instructions to execute.

All processes consist of at least one thread. Multi-threaded processes contain several threads. All threads share the common address space allocated for the process. ...

Sun Microsystems' documentation on threads in its Solaris variant of UNIX (http://www.sun.com/software/whitepapers/solaris9/multithread.pdf, Exhibit H) reads as follows:

Introduction

Multithreading is a popular programming and execution model that allows multiple threads to exist within the context of a single process, sharing the process' resources but able to execute independently. The threaded programming model provides developers with a useful abstraction of concurrent execution. ...

Digital Equipment Corp. (since acquired by Hewlett-Packard) implemented "DECthreads" in its VMS operating system. The January 1999 documentation on DECthreads reads as follows, http://h71000.www7.hp.com/doc/72final/6493/6101pro.html#intro_threads_chap (Exhibit I) (underline added):

1.1 Advantages of Using Threads

Multithreaded programming means organizing and coding a program so that instances of its routines, called threads, can execute concurrently in the same process. You use threads to improve a program's performance – that is, its throughput, computational speed, responsiveness, or some combination.

Using threads can improve a program's performance on uniprocessor systems by permitting the overlap of input, output, or other slow operations with computational operations. Threads are useful in driving slow devices such as disks, networks, terminals, and printers. A multithreaded program can perform other useful work while waiting for the device to produce its next event, such as the completion of a disk transfer or the receipt of a packet from the network.

Even Apple, often the exception to the rule, uses the term "thread" consistently with every other vendor http://developer.apple.com/documentation/Cocoa/Conceptual/Multithreading/index.html#//apple_ref/doc/uid/10000057i (Exhibit J) (underline added):

> **Threads**
>
> In Mac OS X, each process comprises one or more threads. A thread is a stream of execution that runs code for the process. Multiple threads may execute the same code, but they do so independently of other threads....
>
> Threads let your program perform multiple tasks in parallel. For example, you can use threads to perform several, lengthy calculations while your user interface continues to respond to user commands. You could also use threads to divide a large job into several smaller jobs.

A document by David Graves of Hewlett-Packard, http://devresource.hp.com/drc/STK/docs/refs/sol_threads.jsp (Exhibit K), compares and contrasts the major implementations of threads in several operating systems. He notes that "concurrency" is one of the properties that unites all vendors' implementations of threads (underline added):

> **Threads on HP-UX, Solaris, and NT**
>
> This document provides a conceptual mapping of thread function calls between four implementations: HP-UX threads (based on POSIX and X/Open), Solaris threads, POSIX threads on Solaris, and [Microsoft Windows] NT proprietary threads.
>
> ...
>
> **Synchronization**
>
> Since threads run concurrently and share resources, synchronization mechanisms are required to provide mutually exclusive access to shared data. ...

C. The Theory of Operation of "Threads" Was Well-Understood by the Filing Date

Not only was the definition of "thread" well established at the filing date of this application, the theory of operation was so well understood that they were part of the undergraduate curriculum. In particular, the usual implementation involved a series of data structures (sometimes called "control blocks" or "context"), managed either by the operating system or by a run-time library.

For example, Exhibit D, the **freshman-level** slides from UCSD, describes the various attributes of process and threads that must be stored in data structures on a context switch between processes or threads (underline added):

Processes

- Recall that a process includes many things
  - An address space (defining all the code and data pages)
  - . OS resources (e.g., open files) and accounting information
  - Execution state (PC, SP, regs, etc.)
- . Creating a new process is costly because of all of the data structures that must be allocated and initialized ...

The Carnegie-Mellon lecture notes (Exhibit E) describe more of the theory of operation and implementation. In particular, the Carnegie-Mellon notes describe two particular data structures, the "process control block" and "thread control block" that may be used to implement processes and threads (italic and bold in original, underline added):

The Process Control Block (PCB)

We've already discussed several different types of hardware state that are associated with a process. In addition to the hardware-context, there is also the software-context of the process. This includes the state of the programs memory as well as the information that the operating systems maintains about each process. This information is stored in an operating system structure called the *process control block* (pcb). Among other things, the PCB contains the following:

- The *process ID* of the process - a unique number that identifies or names the process within the operating system
- . The group ID - a number that identifies the group or classification of users to which the process belongs.
- . Information about open files
- . Accounting information (CPU time used, bytes read/written, &c)
- . Current state (BLOCKED, READY, RUNNING) (more later)
- . Linked lists and queue pointers (more later)
- . Exit status that is maintained for wait (more later)

...

Threads, a.k.a Light-Weight Processes (LWP)

...

Now suppose that we want multiple process-like things that are separately schedulable -- but share memory. ...

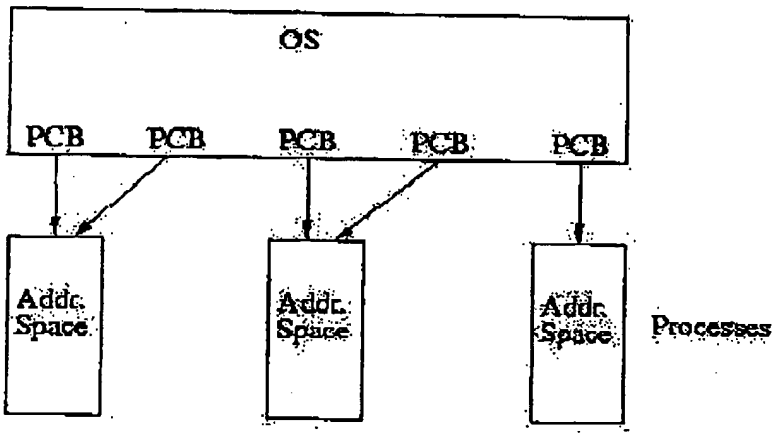
Now consider multiple "processes" sharing the same memory, but otherwise maintaining different state (registers, &c). We call these processes within processes *threads*. We call them this, because they are separate *threads of control* within the same process....

- Fast context switch: This actually depends on the implementation, but switching among threads within the same address space is faster than

switching among processes in different address spaces. We don't need to save and restore the context, including the BASE and LIMIT registers, and other memory-management registers, to context-switch. But depending on how the threads are implemented, the amount of other overhead can vary: it can be very, very fast, or just somewhat faster.

- A special cache, called the TLB doesn't need to be flushed to context switch among threads in the same address space. This not only saves the time it takes to flush the cache, but also maintains the utility of the cache -- this is a big win TLB misses are very expensive.
- A process can do useful work, even while it is blocked: yes, but this also depends on the implementation.

Kernel Supported Threads



We can think of kernel supported threads as a system where multiple PCBs can point to the same address space. Each PCB is really no longer a PCB, but more of a *thread control block (TCB)*. But it is still usually called the PCB.

The PCB now maintains the state of each thread and allows each thread to be scheduled independently. The PCB still holds the usual hardware state, queue state, pointer to address state, and a pointer to the *task control block* (TCB).

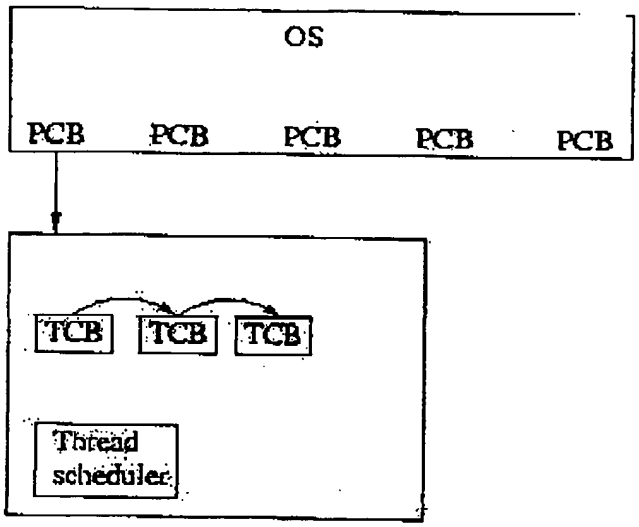
What is a TCB (notice that we are referring to a task control block, not a thread control block)? It maintains the state that is not stored in the PCB -- the state that applies to all threads.

...

In this case the indirection via the TCB allows us to have both common and separate state for the threads.

As the name implies, kernel supported threads require OS support. Many, but not all, OSs support kernel threads.

User-level Threads



Process address space

User-level threads are implemented via a user-level thread library. The kernel neither knows nor cares that they exist. From the kernel's perspective, they are just regular code. User level threads require no changes to the kernel.

TCBs are simply malloc'd each time that a thread is created and linked together to form queues -- within the space of the process.

Context switches among the threads are very cheap -- neither the hardware nor the OS knows or cares about the context of the threads. Since we are not changing address spaces or process state, we are not changing any registers. The state of the threads is just part of the state of the process. This saves much overhead.

The Microsoft Developers' Network Articles explain some of the data structures involved (Exhibit F):

About Processes and Threads

...

A *thread* is the entity within a process that can be scheduled for execution. All threads of a process share its virtual address space and system resources. In addition, each thread maintains exception handlers, a scheduling priority, thread local storage, a unique thread identifier, and a set of structures the system will use to save the thread context until it is scheduled. The *thread context* includes the thread's set of machine registers, the kernel stack, a thread environment block, and a user stack in the address space of the thread's process. Threads can also have their own security context, which can be used for impersonating clients.

The DEC/HP OpenVMS documentation (Exhibit I) describes the data structures used in the theory of operation of processes and threads:

> **6.1 Process Control Blocks (PCBs) and Process Headers (PHDs)**
>
> Two primary data structures exist in the OpenVMS executive that describe the context of a process:
>
> * Software process control block (PCB)
> * Process header (PHD)
>
> The PCB contains fields that identify the process to the system. The PCB comprises contexts that pertain to quotas and limits, scheduling state, privileges, AST queues, and identifiers. In general, any information that must be resident at all times is in the PCB. Therefore, the PCB is allocated from nonpaged pool.
>
> ...
>
> 6.1.1 Effect of a Multithreaded Process on the PCB and PHD
>
> With multiple execution contexts within the same process, the multiple threads of execution all share the same address space but have some independent software and hardware context. This change to a multithreaded process impacts the PCB and PHD structures and any code that references them.
>
> Before the implementation of kernel threads, the PCB contained much context that was per process. With the introduction of multiple threads of execution, much context becomes per thread. To accommodate per-thread context, a new data structure---the kernel thread block (KTB)--- is created, with the per-thread context removed from the PCB. However, the PCB continues to contain context common to all threads, such as quotas and limits. The new per-kernel thread structure contains the scheduling state, priority, and the AST queues.

D. Under the Correct Legal Test, There is No "New Matter"

It is entirely permissible for an amendment to add language to a specification, even though there is no literal support. As the courts have noted, "inherent" support is entirely sufficient. *In re Wright*, 343 F.2d 761, 767, 145 USPQ 182, 188 (CCPA 1965) ("while new language has certainly been added, we are not prone to view all new 'language' ipso facto as 'new matter.' ") The Chisum treatise sets out several classes of new language that are not new matter, Chisum, Patents § 11.04[2][a] (footnotes omitted, underline added):

> Court decisions state, in various ways, that specifications may be amended to "clarify" the original disclosure; thus: (1) "insertion by way of amendment in the description .... of a patent application do not invalidate the patent, if they are only in amplification and explanation of what was already reasonably indicated to be within the invention"; (2) amendments may be made to patent application for the purpose of curing defects, obvious to one skilled in the art, in the ... written descriptions of inventions"; (3) "an amendment to an

> application is not 'new matter' ... unless it discloses 'an invention, process or apparatus not theretofore described.' ... If the later-submitted material accused of being 'new matter' simply clarifies or completes the prior disclosure it cannot be treated as 'new matter' ..." (4) "the amendments to the specification merely render explicit what had been implicitly disclosed originally, and, while new language has certainly been added, we are not prone to view all new 'language' ipso facto as 'new matter.'"

Additionally, the consistent view has been that adding an explicit statement of the ordinary meaning of a term is not "new matter," because that ordinary meaning is inherent in the use of the term. *Ex parte Prince*, 77 USPQ 479, 481 (Pat. Off. Bd. App. 1947); *Ex parte Kharasch*, 67 USPQ 21, 22 (Pat. Off. Bd. App. 1943) (permitting addition of an explicit definition of the term "organic catalyst"); MPEP § 2163.07(I) ("The mere inclusion of dictionary or art recognized definitions ... would not be considered new matter.").

As will be shown below, every sentence of the amendment is either a statement of a legal principle that is an inherent rule of patent interpretation, a definition of a technical term drawn from the art that is inherent in the very use of the term, or an explicit statement of established theories of operation. All of these are well established as not "new matter."

The first sentence objected to is as follows:

> The terms "process" and "thread" are used herein in their ordinary and customary, though formal, senses, as actually used in the programming language systems, operating systems, and processor architecture arts.

This is merely a restatement of the legal test for "broadest reasonable meaning of the words in their ordinary usage as they would be understood by one of ordinary skill in the art." That rule of construction is inherent in every patent application. MPEP §§ 2111, 2111.01. It is not new matter. The next sentence added reads as follows:

> Generally, a "process" is a unit of processor scheduling and protection, each with an associated data structure (or set of data structures) that, in most implementations, holds machine register values and other context associated with the process.

This merely consolidates definitions (6) and (8) of "process" from the IEEE Dictionary (see page 16 of this paper), and the theory of operation for processes described in the UCSD and Carnegie-Mellon lecture notes ("address space," "Execution state (PC, SP, regs, etc.)," "data structures" and "process control block"). Because this sentence merely states the common understanding and theory in the specialized art, previously inherent, this sentence is recognized as not "new

matter." *Wright*, 343 F.2d at 767, 145 USPQ at 188; MPEP § 2163.07(a) ("The application may later be amended to recite the ... theory ... without introducing prohibited new matter.")

> The process data structures, and thus the processes of a computer, are usually under the management of an operating system, usually the operating system's scheduler.

This is merely an explicit statement consolidating the theory of operation, as reflected in the UCSD and Carnegie Mellon lecture notes (exhibits D and E), and the DEC OpenVMS description (Exhibit I) that were previously inherent. This is not new matter. The amendment continues:

> Generally, a "thread" is a flow of control within a process.

This is a direct quote of the ordinary definition of "thread" from the IEEE Dictionary (Exhibit A), and thus is not new matter. *Prince*, 77 USPQ at 481; *Kharasch*, 67 USPQ at 22.

> Each thread has an associated data structure (or set of data structures) that, in most implementations, hold machine register values (usually different than the registers associated with a process) and other context associated with the thread.

This is merely an explicit statement consolidating the definition of "thread (4)" from the IEEE Dictionary, and the theory of operation, as reflected in the UCSD slides and the Carnegie Mellon lecture notes (Exhibits D and E, see discussion "thread control block (TCB)"), and section 6.1.1 of the OpenVMS description (Exhibit I). This is not new matter.

> The thread data structures, and thus the threads of a process, are usually managed either by an operating system or other run time system, to permit the thread to be scheduled independently of and concurrently with other threads of the same process.

This is merely a statement of theory of operation from the Carnegie Mellon lecture notes (sections "Kernel Threads" and "User Level Threads"). It is not "new matter."

The addition to the specification merely reinforces the ordinary meaning in the art, and forecloses reliance on irrelevant and unreasonable definitions that are inconsistent with the specification – definitions that were impermissible before the amendment, and merely become impermissible for an additional reason now. Because the amendment induces no change to the subject matter of the specification or the claims, there is no "new matter."

Statements of ordinary meaning are given special deference when there is no reasonable alternative to the term at issue. *Kharasch*, 67 USPQ at 22 ("It seems impossible to find a term that would be of the exact scope ... It is our view that applicant is entitled ... to employ the simple broad term 'catalyst'" as supplemented by the definition added by amendment). Here, the examiner does not dispute that there is no other term in the art that has the same range of meanings as the formal sense of the term "thread." (Response of April 14, 2005 at 7). If the meaning of the word "thread" is distorted as the examiner proposes, with no alternative word available to overcome the examiner's faulty view, there is no opportunity to claim the an invention that is fully disclosed as required by § 112 ¶¶ 1 and 2.

The examiner's "new matter" objection should be reversed.

E. The Examiner States an Incorrect Legal Test

The examiner acknowledges that the test for "new matter" permits an amendment that was "inherently contained" in the specification as filed. Action of 7/19/05 at 1, lines 14-15. Having conceded that, the examiner then states that he will only allow an amendment that is supported explicitly, by:

> "referenc[ing] those portions of the specification which supported this amended material. Therefore because applicant failed to point to the portions of the specification which support the amended material, there must be no portion of the specification which supports the material (otherwise applicant would have simply pointed to that portion) and therefore, the material is also new matter.

Action of 7/19/05 at 2, lines 1-6.

The examiner is confused. "Inherency" does not require an explicit statement that can be "pointed to;" "inherency" only comes into play when there is no explicit disclosure that could be "referenced." Because the examiner applies an incorrect legal test, the remainder of his reasoning is irrelevant at best. The examiner has made no attempt to show that the amendment is anything other than an explicit statement of material that was previously inherent.

Because the examiner's reasoning is unrelated to the correct legal test, the "objection" should be reversed.

F. The Examiner's Faulty Factual Analysis

It appears that the examiner's primary basis for his "new matter" objection is that the description added by the amendment conflicts with his extrapolated definition. The examiner's extrapolated definition is faulty, and without substantial evidence. *See American Textile Mfrs Institute Inc. v. Donovan*, 452 U.S. 490, 523 (U.S. Sup. Ct. 1981) ("substantial evidence" review requires taking into account evidence that is contrary to the agency's decision).

First, the examiner's definition of "thread" appears nowhere. His statement attributing his definition to the Microsoft dictionary, Action of 7/19/05 at page 3, lines 19-22, is simply false. The examiner relies on pure syllogistic inference, an extrapolation beyond the statements in the Microsoft dictionary. If the amendment agrees with every definition from the relevant arts that is expressly stated, including a definition cited by the examiner himself, and the only "odd man out" is the examiner's extrapolation, then it is the examiner's extrapolation that should be disregarded, not the amendment.

Second, Microsoft itself has acknowledged that the definition of "process" in the 1993 edition of the Microsoft Dictionary, on which the examiner relies, was incorrect – the definition in the current edition is significantly different.

Third, the Microsoft Dictionary was obsolete as of the filing date. Other dictionary editors and standards organizations recognized that the formal definition of "thread" and "process" had changed in the mid-1990's, and by 1996-1998 they withdrew the definitions that had been used in 1993. For example, 1996-2000 definitions of "process," from the IEEE Dictionary (Exhibit A), are as follows:

> **process (3)** An address space and one or more threads of control that execute within that address space, and their required system resources.
>
> **process (6)** An address space with one or more threads executing within that address space, and the required system resources for those threads. ... Many of the system resources defined by this part of ISO/IEC 9945 are shared among all the threads within a process. These include the process ID, the process group ID; the session membership; the real, effective and saved-set user ID; the real, effective and saved-set group ID; the supplementary group IDs; the current working directory; the root directory; the file mode creation mask; and file descriptors.
>
> **process (7)** [essentially similar to (6), in the context of the Ada programming language]

> **process (8)** An address space, a single thread of control that executes within that address space, and its required system resources. On a system that implements threads, a process is redefined to consist of an address space with one or more threads executing within that address space and their required system resources, etc. ...

The relevant definitions changed over this period. For example, the IEEE Dictionary states a definition "process (5)" from the 1993 version of ISO/IEC Std. 9945, the same year as the examiner's edition of the Microsoft Dictionary. However, definition (5) was superseded in a 1996 revision of standard ISO/IEC 9945, by definition (6). Because this application has an effective filing date of 1999, the examiner's 1993 dictionary is irrelevant.

Fourth, the portions of the Microsoft Dictionary on which the examiner relies are clearly informal. Even a cursory comparison of the Microsoft definitions cited by the examiner reveal that they are only informal definitions, barely above the level of a "general usage" dictionary – they reflect none of the characteristics that specialists actually use and rely on. The 1993 Microsoft definition of "thread" states that a thread is merely "part of" a process, without explaining any of the relevant differences and relationships. The 1993 Microsoft definition of "process" never mentions the essential attribute of a "process," its "address space." Microsoft itself discourages precise reliance on its dictionary: both the notes on the back of the current edition of the Dictionary and the Introduction state that the audience for the Microsoft Computer Dictionary is "users" and "home users," but does not mention specialists in programming language systems, operating systems, and processor architecture. The back of the Microsoft Dictionary makes clear that many definitions are over-simplified for non-specialists:

> You get simple, concise definitions for understanding even the most arcane terms..

When Microsoft writes to specialists in the art, and intends for them to rely on its descriptions, Microsoft sets out its definitions much more precisely, as set forth in Exhibit F, and quoted at page 6, above.

Just last month, in a portion of a decision joined by **eleven** of the **twelve judges of the Federal Circuit**, the Court warned against over-reliance on informal definitions, even if those informal definitions are drawn from a technical dictionary:

> For that reason, we have stated that "a general-usage dictionary cannot overcome art-specific evidence of the meaning" of a claim term. ... Even

> technical dictionaries or treatises, under certain circumstances, may suffer from some of these deficiencies. There is no guarantee that a term is used in the same way in a treatise as it would be by the patentee. ...
>
> Finally, the authors of dictionaries or treatises may simplify ideas to communicate them most effectively to the public and may thus choose a meaning that is not pertinent to the understanding of particular claim language. ... The resulting definitions therefore do not necessarily reflect the inventor's goal of distinctly setting forth his invention as a person of ordinary skill in that particular art would understand it.

Phillips v. AWH Corp., 415 F.3d 1303, 1322, 75 USPQ2d 1321, 1334 (Fed. Cir. 2005) (en banc) (underline added). Dictionaries are not infallible.

Fourth, the examiner has twice been challenged to identify any basis to believe that his extrapolation is even correct, let alone "reasonable," or "consistent with the interpretation those skilled in the art would reach." Response of 4/27/05 at 17; Response of 3/30/05 at 6-7. All he cites in support of his extrapolation is his own inference, no direct evidence. By his silence, his inability to provide any evidence in support of his own inference, his failure to rebut the uniform definitions set forth in every other source, and his failure to explain away the absurd consequences of his extrapolation (see Response of 4/27/05 at page 18 and the Response of 3/30/05 at 7), he confesses that his extrapolation is outside the scope of permitted "broadest reasonable interpretation consistent with the specification" and inconsistent with "the meaning given to the term by those of ordinary skill in the art," as required by MPEP § 2111.01.

Thus, any disagreement between the amendment and the examiner's extrapolation is no basis to conclude that the amendment is "new matter." The objection should be reversed.

II. Final Rejection is Premature

Paragraph 10 of the Office Action states that the July 2005 Action is final. This is premature.

A. Premature Final Rejection is a Petitionable Issue

The Federal Circuit[5], the Board of Appeals, and the Director have all held that petition to the Director under Rule 181 is the appropriate avenue request review of premature final

---

[5] *In re Alappat*, 33 F.3d. 1527, 1580, 31 USPQ2d 1545, 1588 (Fed. Cir. 1994) (*en banc*) (Plager, J., concurring) ("The [Director] has an obligation to ensure that all parts of the agency ... conform to official policy of the agency..."); *see also Star Fruits S.N.C. v. United States*, 393 F.3d 1277, 1284-85,

rejection, when finality violates PTO rules, and the only relief requested is reopening of prosecution.

The Board of Patent Appeals and Interferences has long held that premature closing of prosecution is never appealable. *Ex parte Fine*, 217 USPQ 76, 79 (Bd. Pat. App. 1981) (precedential) ("We are likewise not concerned with the allegedly premature nature of the final rejection... This is an administrative matter subject to petition, not a substantive matter within our jurisdiction."); *Ex parte Secor*, http://www.uspto.gov/web/offices/dcom/bpai/decisions/fd981052.pdf (BPAI 2002) (unpublished) (premature final rejection "is reviewable by petition to the Director rather than by appeal to this Board."). Petitioner has diligently sought, and has been unable to find, a single case since 1964 in which the Board reviewed an issue of premature final rejection. There are none. An agency may not require an issue to be presented to a tribunal that "lacks authority to grant the type of relief requested." *McCarthy v. Madigan*, 503 U.S. 140, 147 (1992).

Second, the Board has held that issues arising under MPEP procedures, even those relating to claims, are never appealable – the Board only reviews issues arising under the substantive portions of the Patent Act. *E.g.*, *Ex parte Haas*, 175 USPQ 217, 220 (Bd. Pat. App. 1972) ("If the examiner fails to follow the Commissioner's directions in the M.P.E.P., appellant's remedy is by way of petition to the Commissioner since this Board has no jurisdiction over the examiner's action.") (Lidoff, examiner-in-chief, concurring), *rev'd on other grounds*, 486 F.2d 1053 (CCPA 1973).

Because the issues presented in this petition, and the relief requested, are not appealable, they must be addressed when presented by Petition. 37 C.F.R. § 1.181(a)(1).

Third, the Commissioner of Patents and Trademarks (now the Director) holds that where the sole relief requested is reopening of prosecution, and the rules relied on originate with the Patent Office, instead of the statute – as in this petition – the issues are petitionable, even if the underlying issue might involve some consideration of the merits. *In re Oku*, 25 USPQ2d 1155, 1157 (Comm'r Pats and TM 1992) (emphasis supplied):

---

73 USPQ2d 1409, 1414-15 (Fed. Cir. 2005) ("petition process [is] the 'exclusive administrative check' on the discretion of examiners," to ensure that examiners act within the PTO's rules).

> The designation of a new ground of rejection, while involving a consideration of the merits, also involves the important question of whether the Board followed PTO regulations established by the [Director]....
>
> A decision to reopen prosecution ... is a question solely within the discretion of the [Director] and is in no way a review of a merits decision ...

Finally, issues are petitionable when "the rules specify that the matter is to be ... reviewed by the Director." 37 C.F.R. § 1.181(a)(2). The relevant rule is MPEP § 706.07(c), which instructs as follows:

> **706.07(c) Final Rejection, Premature**
>
> Any question as to prematureness of a final rejection ... is purely a question of practice, wholly distinct from the tenability of the rejection. It may therefore not be advanced as a ground for appeal, or made the basis of complaint before the Board of Patent Appeals and Interferences. It is reviewable by petition under 37 CFR 1.181. See MPEP § 1002.02(c).

The questions presented are within § 1.181(a)(2).

B. Examination Remains Incomplete: An Information Disclosure Statement Has Not Been Considered

The RCE filing of April 27, 2005 requested consideration of the Information Disclosure Statement submitted November 9, 2004. This IDS has not been considered. An application may not be finally rejected while examination is incomplete.

In this and several other applications, Examiner Ellis has repeatedly changed positions, applied references to different claim limitations, reinterpreted claim language, considered IDS's, etc. after "final" rejection, while closing the door against any opportunity to substantively respond to his changed positions. *See* 09/385,394, Petition of April 8, 2005. This is simply wrong, and a violation of the "fairness" that is at the heart of final rejection practice. MPEP § 706.07. Final sauce for the applicant goose is final sauce for the examiner gander. 35 U.S.C. § 3(a)(2)(A) (those who act on behalf of the Director must do so "in a fair, impartial, and equitable manner," that is, with bilateral equity, including with respect to examination under § 131). Examiner Ellis cannot have it both ways. If his work is not complete and timely, and he has to alter his position or complete his work, then prosecution may not be closed against an applicant.

C. The Application Has Not Been Examined – No Issue Has Been Developed For Appeal

In the paragraph spanning pages 2-3 of the Office Action of July 19, 2005, the examiner states that he gives no weight to the amendment because it is new matter. He then states that he examines the claims as if the amendment were already cancelled.

This is not permitted. The MPEP repeatedly instructs that the application must be examined as amended. MPEP §§ 2143.03, MPEP § 2163.06(I), 2163.07.

Even giving the examiner the greatest possible benefit of the doubt on the broadest reasonable definition of the term "thread" in absence of any definition in the specification, MPEP § 2163.07(I) and 2163.06(I) cover the facts here, and instruct that the application must now be examined under the definition stated (underline added, citations omitted):

> Mere rephrasing of a passage does not constitute new matter. Accordingly, a rewording of a passage where the same meaning remains intact is permissible. ... The mere inclusion of dictionary or art recognized definitions known at the time of filing an application would not be considered new matter. If there are multiple definitions for a term and a definition is added to the application, it must be clear from the application as filed that applicant intended a particular definition, in order to avoid an issue of new matter and/or lack of written description. ...
>
> ... The examiner should still consider the subject matter added to the claim in making rejections based on prior art since the new matter rejection may be overcome by applicant.

Here, there is no question that the description of "threads" and "processes" added to the specification reflects the "particular definition" used in the application as filed. Finality should be withdrawn, so that the application can be examined for the first time, giving full weight to the definition stated in the specification and the "broadest reasonable meaning of the words in their ordinary usage as they would be understood by one of ordinary skill in the art," as expressed in every relevant dictionary. MPEP § 2111.01.

On the record as it stands, no clear issue is developed for appeal – are the claims to be interpreted on appeal "consistently with the specification," or under the examiner's extrapolation, with the amendment ignored? Final rejection is premature. MPEP § 706.07.

When an agency employee acts short of "requirements of the applicable departmental regulations," then employee's action is "illegal and of no effect." *Vitarelli v. Seaton*, 359 U.S. 535, 545 (1959); *IMS, P.C. v Alvarez*, 129 F.3d 618, 621 (D.C. Cir. 1997) (it is a "well-settled

rule that an agency's failure to follow its own regulations is fatal to the deviant action."). The examiner's disregard of the amendment to the specification departs from agency procedures. Thus, as a matter of law, there is no rejection of this application, let alone final. Finality should be withdrawn so that the examiner will have an opportunity to examine the application.

III. Conclusion

The "new matter" objection should be reversed. "Finality" of the Office Action of July 19, 2005 should be reversed, and the examiner instructed to apply the definition set forth in the specification and the dictionaries cited above. (To avoid any doubt, this Petition does not request any adjudication on the ultimate merits of any claim, only the failure of the examiner to comply with procedural requirements for examination.)

This petition occasions no fee. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-28-000053BS.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: September 15, 2005

By: _____

David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8757
(212) 728-9757 Fax